Firewall

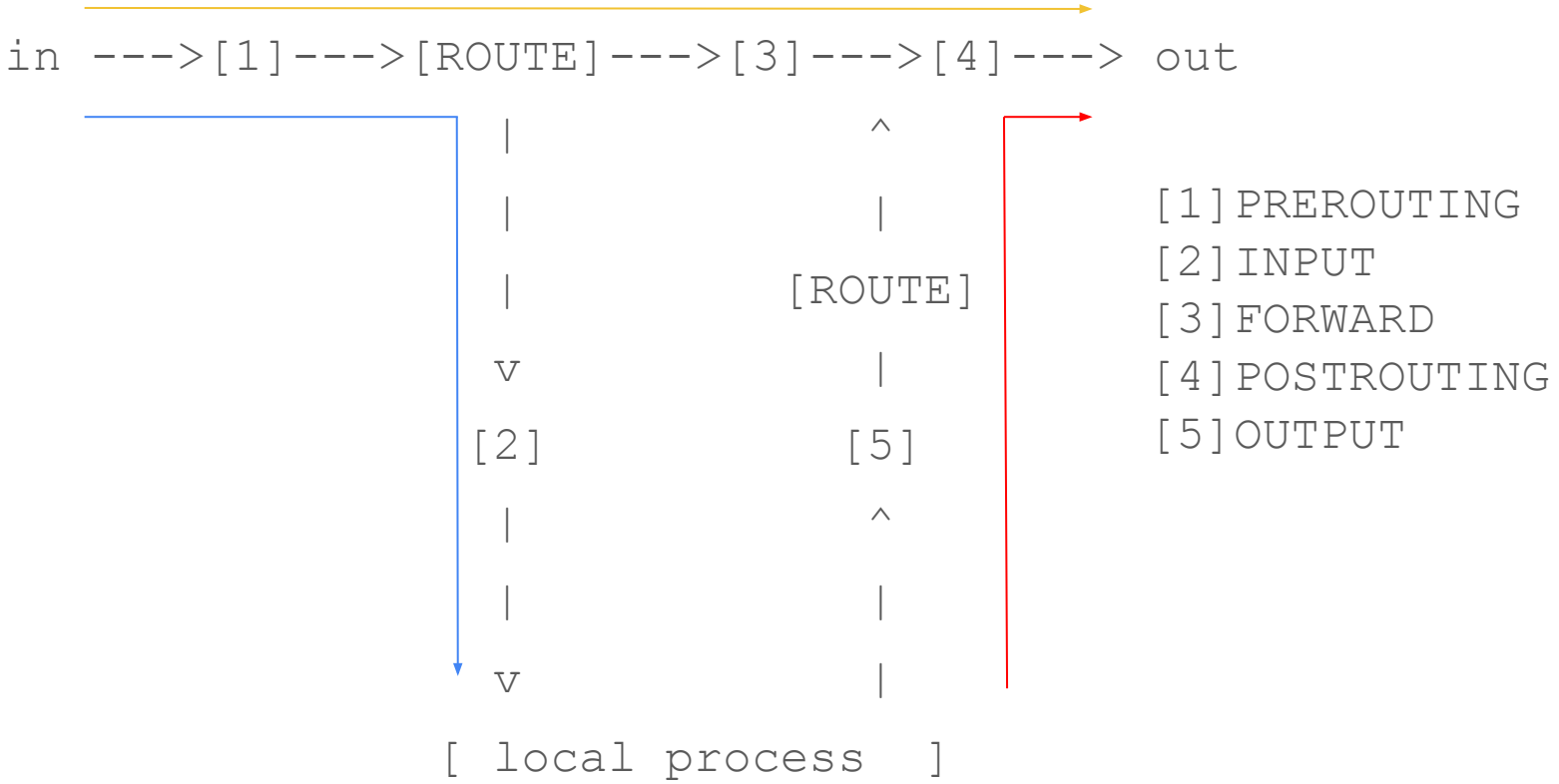
Netfliter - overview

The netfilter project enables packet filtering, network address [and port] translation (NA[P]T), packet logging, userspace packet queueing and other packet mangling.

Netfliter - components

- Hooks in a packet's traversal of the protocol stack
- conntrack and NAT kernel modules
- userspace libraries for userspace packet handling
- iptables (ip6tables, arptables, ebtables)
- nftables - replacement for {ip/ip6/arp/eb}tables

Netfilter - hooks



Netfliter - conntrack module

- conntrack is a part of kernel that is responsible for tracking connections. Please note that from conntrack POV connectionless protocols like UDP also form connections. Connection for conntrack in case of e.g. UDP is a pair of tuples. One (src_ip, src_port, dst_ip and dst_port) and a second one for the reply direction.
- contrack has 5 states for connections:
    - NEW - only one valid packet traveled in the original direction
    - ESTABLISHED - the valid reply packet was observed
    - RELATED - the connection is related to the other one. E.g. icmp error code can be related to the failed TCP connection
    - INVALID - not matching any of definitions above
- There is also conntrack userspace tool to interact with it

Netfliter - nat module

- NAT module is a part of kernel responsible for applying NAT transformations to the packets
- It needs the information from the conntrack to know how and when to transform the packet
- Only the first packet from every connection reaches nat rules as defined by iptables/nftables. The rule is responsible only for configuring conntrack and nat modules to apply appropriate transformations to all other packets for this connection.

Netfliter - userspace libraries

- netfilter framework allows for queuing packets. It is mostly used to handle them asynchronously in user-space.
- To get the packets from the queue libnetfilter_queue can be used
- libnetfilter_conntrack can be used to manipulate conntrack entries during userspace processing e.g. for enabling nat.

Netfliter - iptables

- iptables (ip6tables, arptables, ebtables) are the userspace tools used to configure firewall/nat rules. Those rules are the main building blocks of the firewall configuration.
- Rules are stored together and executed in chains which are hooked into the netfilter hooks.
- Chains of the same type/use case are stored together inside 5 predefined tables: filter, nat, raw, mangle, security.

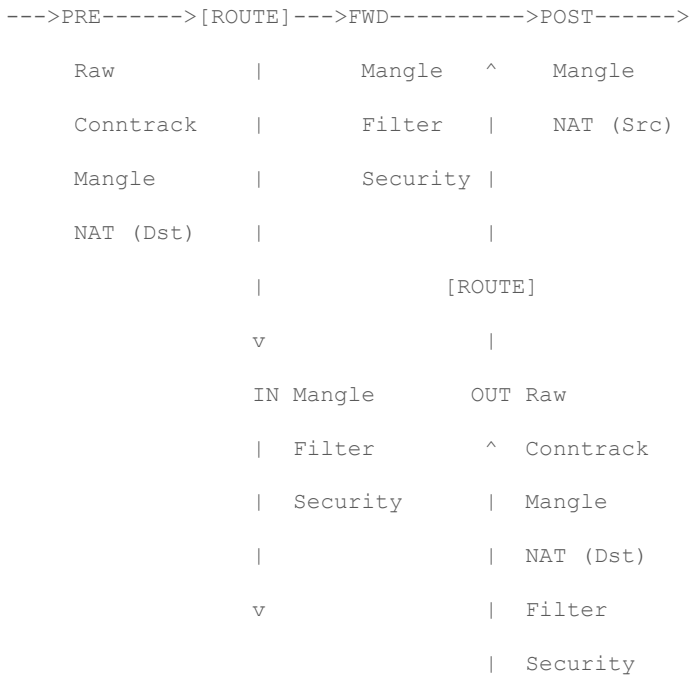
Netfliter - nftables

- nftables is the new, much improved version of iptables
- single tool in place of ip/ip6/arp/eb tables and ipset
- faster
- flowtables for fast bypasses
- more flexibility:
    - no predefined chains or tables
    - arbitrary number of user-defined tables
    - base chains’ hooks and priorities are configurable

Hooks

- Kernel modules can register to listen at any of these hooks
- A module that registers a function must specify the priority of the function within the hook
- Each module registered at that point is called in the order of priorities, and is free to manipulate the packet
- The module can then tell netfilter to do one of five things:
  - **NF_ACCEPT**: continue traversal as normal.
  - **NF_DROP**: drop the packet; don't continue traversal.
  - NF_STOLEN: I've taken over the packet; don't continue traversal.
  - **NF_QUEUE**: queue the packet (usually for userspace handling).
  - NF_REPEAT: call this hook again.

iptables built-in tables and chains

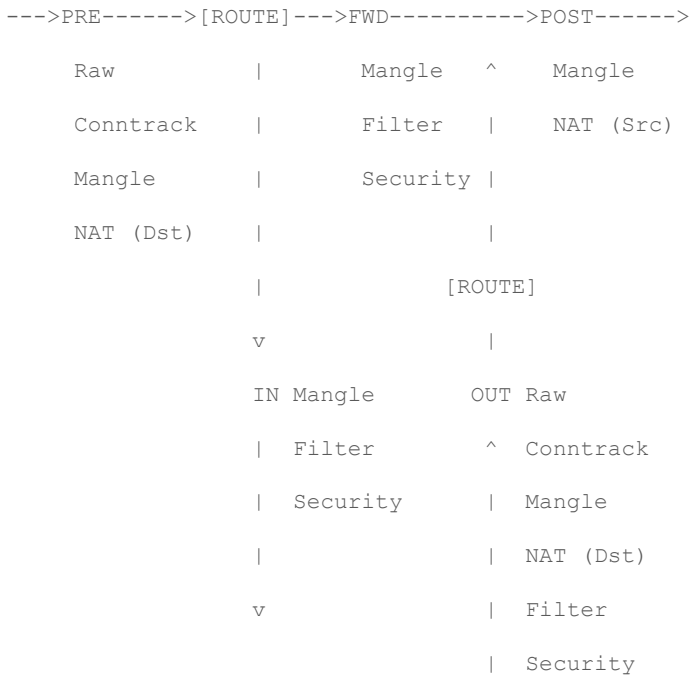


- Rules in iptables are organized by tables
- Iptables defines 5 tables: filter, nat, mangle, raw, security
- These tables classify rules according to the type of decisions they are used to make

iptables built-in tables and chains

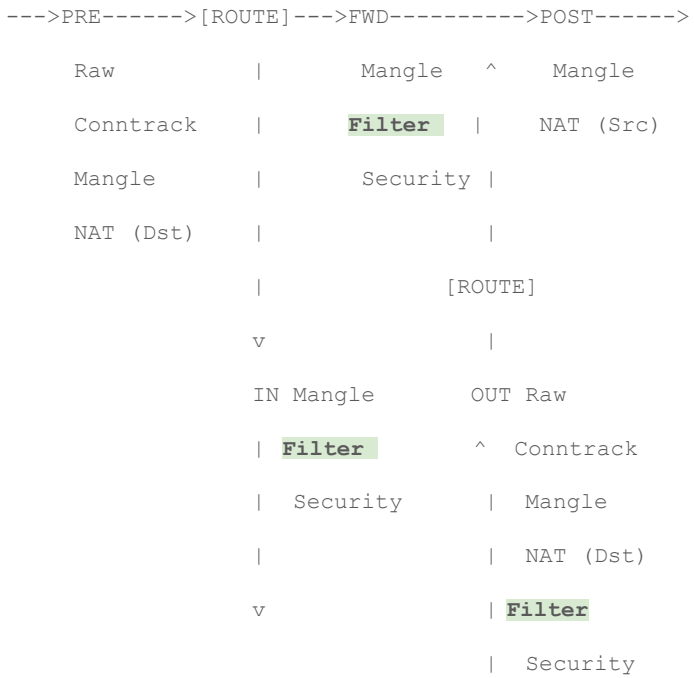
- filter - default table. rules from this table should be responsible for filtering the packets that is either accepting or dropping.
- nat - rules from this table are special. Only first packet from the connection is sent to them. They are responsible only for configuring NAT and later conntrack and nat frameworks handle the natting for the rest of the packets.
- mangle - rules from this table should be used for mangling the packets. E.g. TTL could be adjusted here.
- raw - rules from this table have the highest priority. Even higher then conntrack. They’re actually only base hooks called before conntrack and their main purpose is to mark packets with NOTRACK target to disable connection tracking.
- security - for Mandatory Access Control (MAC) networking rules in SELinux

iptables built-in tables and chains



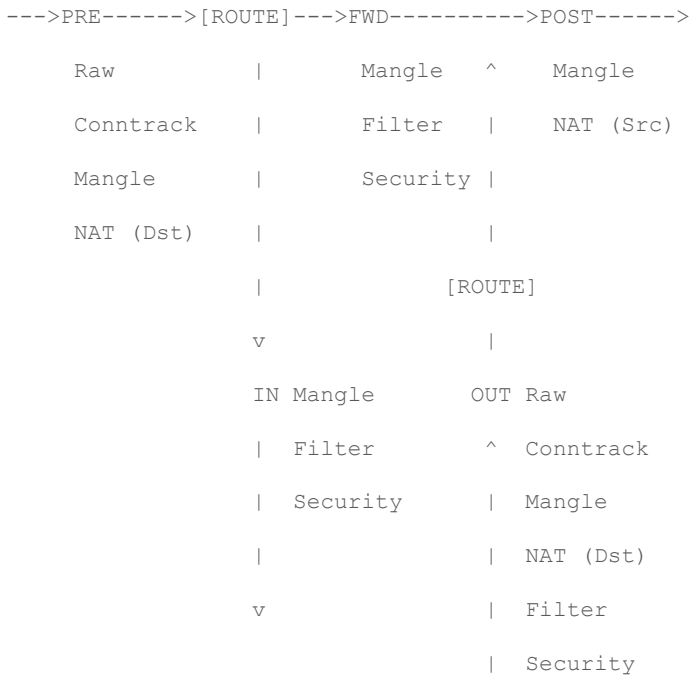
- Within those tables rules are kept inside chains
- While tables are defined by the general aim of the rules they hold, the chains represent actual hooks that trigger them
- Chains determine when rules will be evaluated

iptables built-in tables and chains



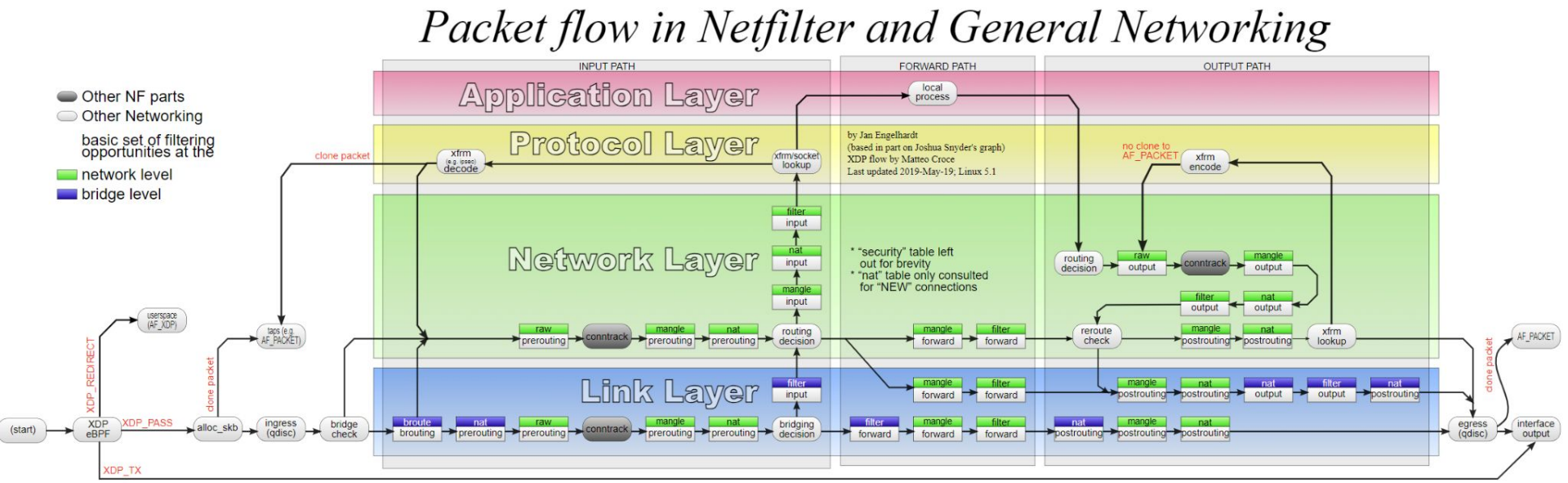
- iptables have built-in base chains
- Those are the chains that are called directly by netfilter framework.
- They can call other user-defined chains
- Built-in chains have their priorities predefined
- Built-in chains must have a policy defined. That is a default target for the packet in case no rules matched the packet. Either ACCEPT or DROP

iptables built-in tables and chains



Tables↓/Chains→	PREROUTING	INPUT	FORWARD	OUTPUT	POSTROUTING
(routing decision)				✓	
raw	✓			✓	
(connection tracking enabled)	✓			✓	
mangle	✓	✓	✓	✓	✓
nat (DNAT)	✓			✓	
(routing decision)	✓			✓	
filter		✓	✓	✓	
security		✓	✓	✓	
nat (SNAT)		✓			✓

iptables packet flow



nftables

- nftables have different approach to tables, chains and address families.
- nftables handles all address families: ip, ip6, inet (ip/ip6), arp, bridge (L2), netdev (network device - hooks right after network taps e.g. tcpdump)
- There are no built-in tables. Tables are containers for chains. They are identified by the their address family and their name.
- There are no built-in chains. Chains are containers for rules. There are two types of chains. Base chains and regular chains. Base chains are called directly from netfilter hooks. When base chain is defined it must have specified: type, hook and priority.

nftables - types of chains

Type	Families	Hooks	Description
filter	all	all	Standard chain type to use in doubt.
nat	ip, ip6, inet	prerouting, input, output, postrouting	Chains of this type perform Native Address Translation based on conntrack entries. Only the first packet of a connection actually traverses this chain - its rules usually define details of the created conntrack entry (NAT statements for instance).
route	ip, ip6	output	If a packet has traversed a chain of this type and is about to be accepted, a new route lookup is performed if relevant parts of the IP header have changed. This allows to e.g. implement policy routing selectors in nftables.

nftables - priorities

Name	Value	Families	Hooks
raw	-300	ip, ip6, inet	all
mangle	-150	ip, ip6, inet	all
dstnat	-100	ip, ip6, inet	prerouting
filter	0	ip, ip6, inet, arp, netdev	all
security	50	ip, ip6, inet	all
srcnat	100	ip, ip6, inet	postrouting

- Standard priority values can be replaced with easily memorizable names. Not all names make sense in every family with every hook as shown in the table but their numerical value can still be used for prioritizing chains.
- These names and values are defined and made available based on what priorities are used by xtables when registering their default chains.

nftables - priorities

Name	Value	Families	Hooks
raw	-300	ip, ip6, inet	all
mangle	-150	ip, ip6, inet	all
dstnat	-100	ip, ip6, inet	prerouting
filter	0	ip, ip6, inet, arp, netdev	all
security	50	ip, ip6, inet	all
srcnat	100	ip, ip6, inet	postrouting

- Standard priority values can be replaced with easily memorizable names. Not all names make sense in every family with every hook as shown in the table but their numerical value can still be used for prioritizing chains.
- These names and values are defined and made available based on what priorities are used by xtables when registering their default chains.

iptables / nftables situation

- nftables are flexible enough to implement all iptables functionalities
- Because of that on modern OSs iptables tools are just wrappers around nftables. You can see it when running iptables –version. iptables v1.8.7 (nf_tables) will be printed if nftables backand is used. In other case it will print iptables v1.8.7 (legacy)
- You can still use iptables syntax but the rules are stored in nftables. You can see it by adding rules via iptables and printing them via nftables
- iptables-translate tool is useful to translate rules from iptables to nftables syntax

Rules

- Rules in chain are evaluated top to bottom
- The rule has two parts. First part is matching statement. Packet is matched to the defined rules.
- If the rule does not match the next rule is executed. If no rule was matched the verdict will be decided by the chain policy.
- If the rule does match the second part is executed. Second part is the target statement. It defines the action to be performed on the packet. There are two types of targets. Terminating and non-terminating. Terminating targets e.g. ACCEPT, DROP, REJECT, RETURN stop the evaluation of other rules in the chain. Non-terminating targets e.g. LOG perform the action and allow execution of other rules in the chain.
- Target statement can also jump to other user defined chains. In such case RETURN means finish currently processed chain and continue execution of the previous chain that jumped to the current one.

iptables - syntax

```
iptables -t TABLE_NAME COMMAND CHAIN …

TABLE_NAME:
  - filter (default)
  - nat
  - mangle
  - raw
  - security

CHAIN:
  - PREROUTING
  - INPUT
  - FORWARD
  - OUTPUT
  - POSTROUTING

COMMAND:
  -A - append
  -C - check
  -D - delete
  -L - list
  -F - flush (remove all rules)
  -N - new chain
  -X - delete chain
  -P - set policy for the built-in chain
```

iptables - syntax

```
iptables -t TABLE_NAME -A CHAIN RULE_SPECIFICATION

RULE_SPECIFICATION
  -p - protocol (tcp, udp, icmp, …)
  -s - source ip
  -d - destination ip
  -i - input interface
  -o - output interface
  -m - match (use more sophisticated match from iptables-extensions)
  -j - jump to target (either basic verdict like e.g. DROP or target extension like
       e.g. NFQUEUE or user-defined chain)
```

Rules - quick examples

Drop all tcp packets coming to this machine from 10.0.1.3 ip source:

```
iptables -A INPUT -p tcp -s 10.0.1.3 -j DROP
```

```
nft add rule ip filter INPUT ip protocol tcp ip saddr 10.0.1.3 counter drop
```

Accept incoming packets only if the connections is established (that means that we needed to initiate it)

```
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
nft add rule ip filter INPUT ct state related,established counter accept
```

Enable Source NAT on the eth1 interface:

```
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 10.2.4.5
```

```
nft add rule ip nat POSTROUTING oifname "eth1" counter snat to 10.2.4.5
```

Rules - more examples

- There is huge amount of extensions that makes it impossible to present them all. For more match extensions and target extensions look into

```
man iptables

man iptables-extensions

man nftables
```